

**Notes around the GPLv3  
further to the meeting between Eben Moglen and OW2**

**Cedric Thomas  
OW2 Consortium  
June 2008**

The opinions expressed in this paper are those of the author and do not necessarily reflect the views of OW2 Consortium.



## Introduction

Eben Moglen ([http://en.wikipedia.org/wiki/Eben\\_Moglen](http://en.wikipedia.org/wiki/Eben_Moglen) and <http://moglen.law.columbia.edu/>) has been with the Free Software Foundation serving as general counsel since 1994 and board member from 2000 to 2007 where he became heavily involved with drafting version 3 of the GPL. Early June, this year, he held a series of meetings in Europe with the double objective apparently to promote the GPLv3 and his newly established law practice. Indeed the man is a true advocate of free software and is at home with IP law; he deserves his broad recognition.

The meeting touched on several key IP issues relevant to the OW2 community. Here is a short synthesis based on my own notes, those of Pierre-Yves Gibello ([gibello.com/blog](http://gibello.com/blog)) and Jean-Marie Chauvet (<http://www.itrmanager.com/tribune/202/eben-moglen-moderne-ouvrier-trole-jean-marie-chauvet.html>) and a debriefing conference call with Pierre-Yves and Emmanuel Paret of Orange Group. In the text below, quotes ("...") denote verbatim either for the GPL text or from Eben Moglen comments.

## Why the GPLv3

The GPLv3 was drafted to overcome some limitations of the v2, specifically its connection with US law. Presumably, the v2 might not resist a strong, and meticulous attack in a non-US court. The GPLv3 is meant to be a transnational legal instrument.

The key idea behind it is to define "how to give a permission that works in all legal systems in a more or less predictable way."

The starting point of the license is a statement of fact: Something has been done that requires permission under a copyright law. Two things are important here: the definition of what is done that requires permission and the applicable copyright law.

## **Applicable copyright law**

With the GPLv3 the interpretation of the license does not depend on the US law but on the local (often national) copyright law. The applicable copyright law is thus determined case by case by the legal context. Interpretations can differ for instance in the definition of "linkage" between two pieces of code.

The GPLv3 does not provide answers but it indicates which is the legal framework, i.e. the « applicable copyright law », within which situations must be analysed and questions must be answered.

## **Propagation**

At the beginning, there is propagation. The license is applicable only if there is propagation, but "there is propagation only if you do something that requires permission under local copyright law".

Therefore, the GPLv3 allows modification by users for their own use. Modification for use within one entity (i.e. throughout a network of local branches), is not propagation. In the same way, a modification performed by a sub-contractor (a systems integrator) for the private usage of the contractor is not propagation.

Propagation is also defined by what technically happens to the code as explained in the next paragraph.

## **Derivative work, etc.**

What is the perimeter of the work which is actually covered by the GPLv3? The GPLv2 refers several times to derivative work, however, the definition of a derivative work raises questions (...) this is why the word derivative does not appear in the v3 and is replaced by the expression ("work based on").

The GPLv2 basically states that if you "distribute two things together they become one work.") whereas the GPLv3 says the same thing but depending on the local copyright law.

Therefore, there are situations where lawyers will have to decide if what happened to the code requires permission under their applicable copyright law. This is where Eben Moglen comments provide some guidance.

If a piece of code is modified, then the license must be conveyed when the code is shared, i.e. the modified code can only be shared with the same license. This is straightforward.

Linkage is a more complicated situation. Typically, in the case of linkage which creates larger works which can be concerned by local copyright law. According to Eben Moglen, the license applies to static linkage but not dynamic linkage:

- Static linkage: binding two pieces of code by compiling them together like taking a chapter from somebody else's book and inserting it in your own book,
- Dynamic linkage: like over a network where linkage happens at run time, the two pieces of code are not distributed together, this is NOT propagation

The question is still open if linking Java classes at run time is considered static linkage or dynamic linkage...

As Eben Moglen puts it: "The issue is how do the parts relate to one another? Can they operate independantly?" The GPL license is only conveyed to the piece of code that cannot operate that specific GPLed code. Only in that case can the following sentence of the GPLv3 apply "The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work."

With Eben Moglen, we agreed on some simple exemples:

- A business application running with the MySQL database (which is covered by the GPL) is not a "work based on" MySQL because it depends on a standard interface (the SQL language) and it could run comparably on another SQL database.

- A portlet running in a GPL portal through standard interfaces does not become GPL if it can run on another portal offering the same interfaces

Pierre-Yves Gibello suggested a precise exemple: a servlet deployed in a GPL container. Here the analysis is that the servlet does not become GPL if it uses standard APIs (i.e. Aps which are not *unique* to that container) and if there is a non-GPLed container available on the market in which it can run comparably. Therefore, a component does not become GPL if both conditions are met: interchangeable APIs (the component use only public APIs) and interchangeable program (the component is not dependent on that particular program).

## **LGPL and Affero**

Eben Moglen introduces the LGPLv3 and Affero GPLv3 as two extensions to the GPLv3. The LGPLv3 provide additional permission whereas the AGPLv3 an additional restriction:

- LGPLv3: additional permission to link (dynamically or statically) with proprietary code.
- AGPLv3: additional restriction or, rather, obligation to provide the source code somewhere.

More specifically, with the LGPLv3, "you can use LGPL code in your program as long as the recipient keeps all the LGPL rights on the LGPLed code." In other word, the LGPL code used with a proprietary code remains LGPL.

As a result of the discussion supra on dynamic linkage, it appears that AGPLv3 only applies to program, or platforms, with interfaces which are unique to this program (in other words in the case where the quality of the interface bring specific value to the code using these interfaces that no other interfaces would bring).

## **On embedded software**

With IT and middleware increasingly concerned by consumer usage and incorporated into

consumer devices, the issue of open source embedded software (also referred as Tivoization) becomes of interest for the OW2 community.

Unlike other licenses, the GPLv3 has no provision regarding combination with hardware. For instance, the Eclipse Public License (EPL) says "The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder." In other words, the EPL is only application to the combination of a Program and a Contribution (terms defined by the license) and hardware implementations are not concerned. Eben Moglen explains that this kind of provision derives from the efforts of Sun Microsystems to protect its royalties revenues from Java licenses for mobile phones.

The GPLv3 ambitions to impact hardware implementations because, as Eben Moglen says "the risk is that all code would eventually be locked down into a device and nothing would be programmable anymore." Therefore, the GPLv3 give access to embedded GPL program by ensuring that:

- a) "nobody is a criminal for having modified a GPL program"
- b) "if you put GPLv3 software in a consumer device, you have to answer how to install and modify versions of the code"

Jean-Marie Chauvet highlights however that this good intention is based on a US definition of consumer devices which excludes warfare systems, secure defense networks, medical equipment, etc. and questions if this definition is valid outside USA.

### **On free licenses and dual licensing**

Eben Moglen explains that the choice of a free license by a developer (an individual or a company) for a given software is a way to generate a community around this software. Moreover it provides a strategic advantage because it is a community that no-one can take over. More generally, free software licenses are valuable for the whole society by helping support "non-profit supply chains" which benefit everyone who play by the rule of giving back

to the community.

Dual licensing is a way to discriminate between users who play by the rule and who are comfortable with the free software license and users who behave like mere consumers without returning anything to the community: the later can legitimately be pushed toward a commercial license. Dual licending is a sensitive issue because it implicitly generates FUD (fear, uncertainty and doubt) in the mind of users who might as a consequence consider free software as non viable.

## **Conclusion**

The Free Software Foundation and the GPL have triggered a true paradigm shift in the software industry. This trend is carried out in different manners by dozens of open source licenses. Although the GPLv3 can be seen as a reference milestone it is but one license among others. However, by referring to local applicable copyright laws, the GPLv3 is a key attempt at providing a transnational legal framework to the open source movement, it also highlights the growing need for competent lawyers in this increasingly complex field.